Chapter 3:

Boolean Algebra & Logic Gates

Binary Logic

If I move the switch on the wall up, the light will turn on."

- At first glance, this seems to be a correct statement. However, if we look at a few other factors, we realize that there's more to it than this. A more complete statement would be:

"If I move the switch on the wall up *and* the light bulb is good *and* the power is on, the light will turn on."

If I move the switch on the wall up, the light will turn on."

Light = Switch

This means nothing more than that the light will follow the action of the switch, so that when the switch is ***up/on/true/1*** the light will also be ***on/true/1***. Conversely, if the switch is ***down/off/false/0*** the light will also be ***off/false/0.***

"If I move the switch on the wall up *and* the light bulb is good *and* the power is on, the light will turn on."

Light = Switch *and* Bulb *and* Power

Normally, we use symbols rather than words to designate the *and* function that we're using to combine the separate variables of Switch, Bulb, and Power in this expression. The symbol normally used is a dot, which is the same symbol used for multiplication in some mathematical expressions.

Light = Switch · Bulb · Power

Binary logic = binary variables + set of logical operations

- Variables
  - Designated by letters of the alphabet, such as *A*, *B*, *C*, *x*, *y*, *z*, etc.,
  - Each variable has two and only two distinct possible values: 1 and 0.

- 3 basic logical operations:
  - AND,
  - OR, and
  - NOT.

- Each operation produces a binary result, denoted by *z*.

Why Logic Gates

- *Logic gates* - Electronic circuits which combine digital signals according to the Boolean algebra

- Gates because they control the flow of information.
  - *Positive logic* is an electronic representation in which the true state is at a higher voltage.
  - *Negative logic* has the true state at a lower voltage.

- In digital circuits all inputs must be connected.

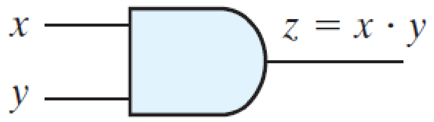
The AND Gate

- Represented by a dot

E.g.

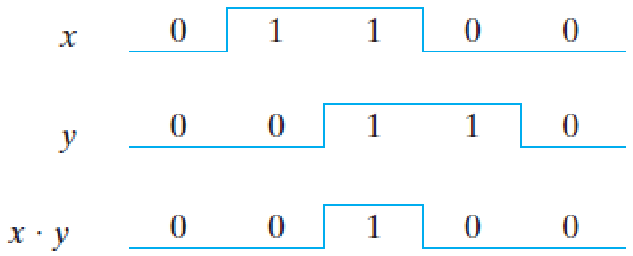
$x \cdot y = z$ *Read as* x AND y is equal to z

- $z = 1$ if and only if $x = 1$ and $y = 1$; otherwise $z = 0$.



x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

truth table



The OR Gate

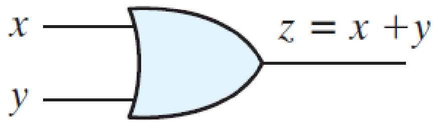
- Represented by a plus (+) sign.

E.g

$x + y = z$ is read as x OR y is equal to z

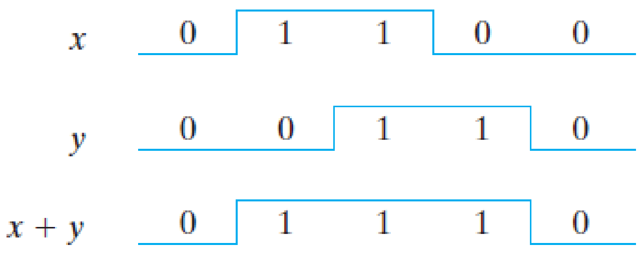
meaning that

- $z = 1$ if $x = 1$ or if $y = 1$ or if both $x = 1$ and $y = 1$.
- If both $x = 0$ and $y = 0$, then $z = 0$.



x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

truth table



The NOT gate (complement operation/Inverter)

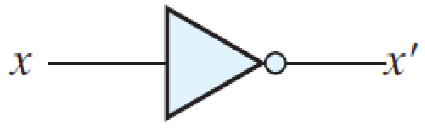
- Represented by a prime (') (sometimes by an overbar).

E.g.

$x' = z$ (or $\overline{x} = z$) is read as not x is equal to z,

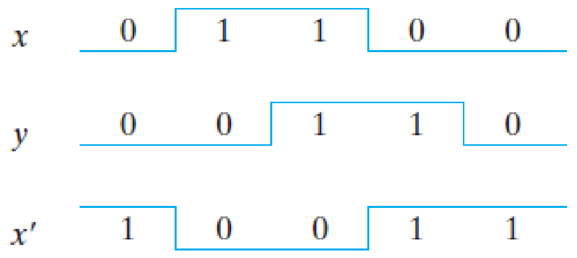
meaning that z is what x is not.

if $x = 1$, then $z = 0$, but if $x = 0$, then $z = 1$.



x	x'
0	1
1	0

truth table



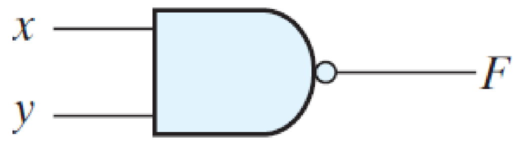
The "bubble" (o) present at the end of the NOT gate symbol above denotes a signal inversion (complementation) of the output signal.

Derived Logical Functions and Gates

- While the three basic functions AND, OR, and NOT are sufficient to accomplish all possible logical functions and operations, some combinations are used so commonly that they have been given names and logic symbols of their own:
  - NAND,
  - NOR,
  - XOR.
  - Exclusive-NOR

The NAND Gate

- NAND function is inverted AND function
- Represented by a (.)prime (.)' or (.)overbar



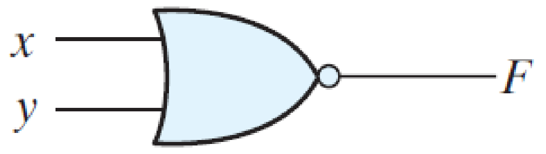
$F = (xy)'$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

The NOR Gate

- The NOR gate is an OR gate with the output inverted.

- Represented by a (+)prime (+)' or (+)overbar

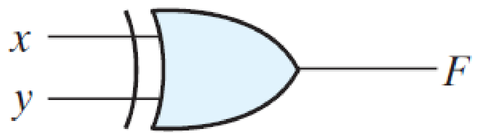


$F = (x + y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

The Exclusive-OR, or XOR Gate

- Represented by a plus (+) sign with a circle around it.
- "Either x or y, but not both."
- The XOR gate produces a logic 1 output only if its two inputs are *different*. If the inputs are the same, the output is a logic 0.



$$F = xy' + x'y$$
$$= x \oplus y$$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

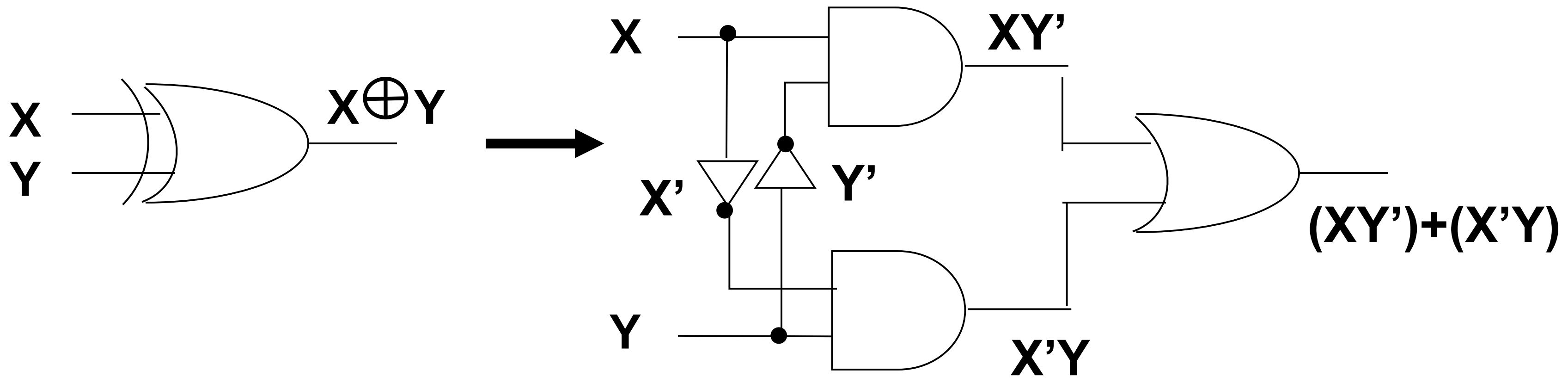
- XOR Theorem: For each X, Y in B,

$$X \oplus Y = (X \,.\, Y') + (X' \,.\, Y)$$

In words,

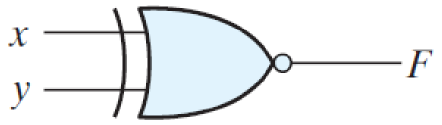
X xor Y = (X and (not Y)) or ((not X) and Y)

And, as an equivalent circuit,



The Exclusive-NOR (equivalence)

- This is an Exclusive –OR gate with an inverter at the output



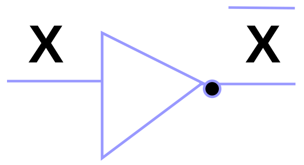
$$F = xy + x'y' = (x \oplus y)'$$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

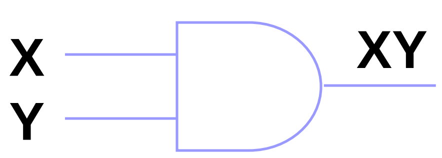
Summary

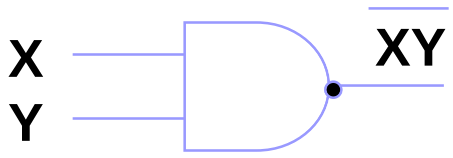
- Each logic operation has its own gate diagram and truth table:

not (inverter)

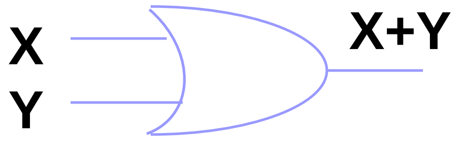


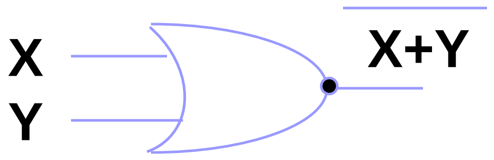
and / nand



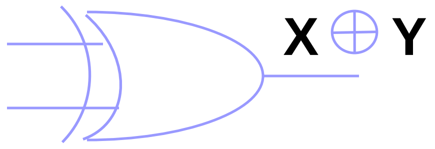


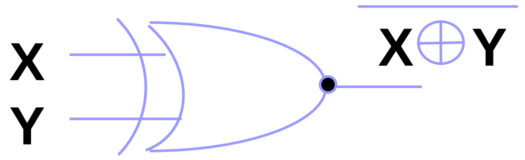
or / nor





xor / xnor





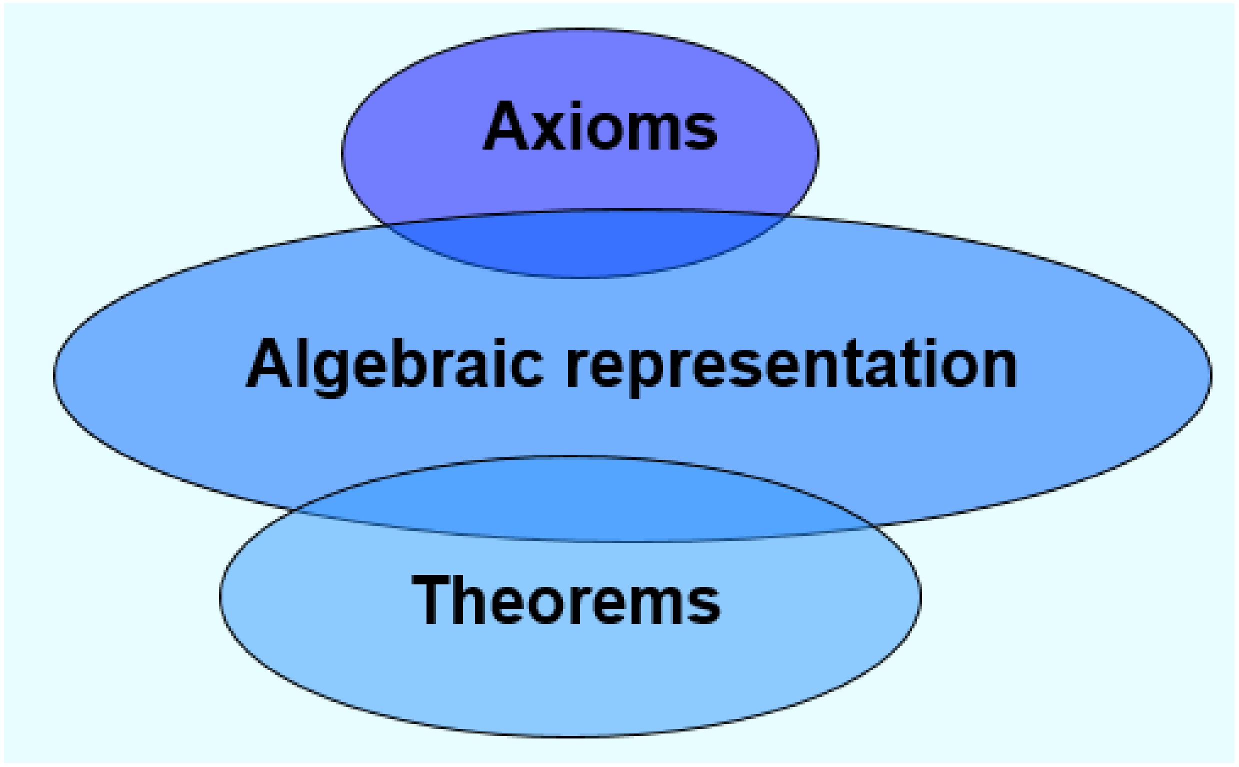
Boolean Algebra

- Used to express logic functions algebraically.
- Defined with
  - a set of elements,
  - a set of operators,
  - a number of unproved axioms or postulates.

- A *set* of elements is any collection of objects having a common property. E.g.
  - If S is a set and *x* and *y* are certain objects, then x∈S and *y*∈S denotes that *x and y* is a member of the set S
  - A = {1,2,3,4}, *i.e.* the elements of set A are the numbers 1, 2, 3, and 4.

- Axioms or Postulates of Boolean algebra are a set of logical expressions that we accept without proof and upon which we can build a set of useful theorems.

	AND Operation	**OR Operation**	**NOT Operation**
Axiom 1	$0 \cdot 0 = 0$	$0+0 = 0$	$0 = 1$
Axiom 2	$0 \cdot 1 = 0$	$0+1 = 1$	$1 = 0$
Axiom 3	$1 \cdot 0 = 0$	$1+0 = 1$	
Axiom 4	$1 \cdot 1 = 1$	$1+1 = 1$	


Axioms
Algebraic representation
Theorems

The most common postulates

1.	*Associative law.*	$(x * y) * z = x * (y * z)\, for\, all\, x, y, z, \in S$
2.	*Commutative law.*	$x * y = y * x\, for\, all\, x, y \in S$
3.	*Identity element.*	$e * x = x * e = x\, for\, every\, x \in S$
4.	*Inverse.*	$x * y = e$
5.	*Distributive law.*	$x * (y \bullet z) = (x * y) \bullet (x * z)$
6.	*Complementarity:*	$a + a' = 1 a \bullet a' = 0$

- *A field is a set of elements, together with two binary operators, each having properties 1 through 5 and both operators combining to give property 6.*

- The *distributive* law $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ can be shown to hold from the operator tables by forming a truth table of all possible values of *x*, *y*, and *z*.

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$y + z$	$x \cdot (y + z)$
0	0
1	0
1	0
1	0
0	0
1	1
1	1
1	1

$x \cdot y$	$x \cdot z$	$(x \cdot y) + (x \cdot z)$
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	1	1
1	0	1
1	1	1

Duality Property

- Duality states that *"Every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged"*.
  - All Boolean expressions have logical duals
  - Any theorem that can be proved is also proved for its dual
  - Replace: • with +, + with •, 0 with 1, and 1 with 0
  - Leave the variables unchanged

- If the *dual* of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

Basic Theorems

- de Morgan’s Theorem
  - Procedure for complementing Boolean functions
  - Replace: • with +, + with •, 0 with 1, and 1 with 0
  - Replace all variables with their complements

Postulates and Theorems of Boolean Algebra

	(a)	(b)
Postulate 2	$x + 0 = x$	$x \bullet 1 = x$
Postulate 5	$x + x' = 1$	$x \bullet x' = 0$
Theorem 1	$x + x = x$	$x \bullet x = x$
Theorem 2	$x+1=1$	$x \bullet 0 = 0$
Theorem 3, involution	$(x')' = x$	
Postulate 3, commutative	$x + y = y + x$	$xy = yx$
Theorem 4, associative	$x + (y + z) = (x + y) + z$	$x(yz) = (xy)z$
Postulate 4, distributive	$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	$(x + y)' = x'y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x(x + y) = x$

Theorem 1(a):

$x + x = x.$

Statement	Justification
$x + x = (x + x) \cdot 1$	postulate 2(b)
$= (x + x)(x + x')$	5(a)
$= x + xx'$	4(b)
$= x + 0$	5(b)
$= x$	2(a)

Theorem 1(b):

$x \cdot x = x.$

Statement	Justification
$x \cdot x = xx + 0$	postulate 2(a)
$= xx + xx'$	5(b)
$= x(x + x')$	4(a)
$= x \cdot 1$	5(a)
$= x$	2(b)

- Theorem 1(b) is the dual of theorem 1(a)

Theorem 2(a): $x + 1 = 1.$

Statement	**Justification**
$x + 1 = 1 \cdot (x + 1)$	postulate 2(b)
$= (x + x')(x + 1)$	5(a)
$= x + x' \cdot 1$	4(b)
$= x + x'$	2(b)
$= 1$	5(a)

Theorem 2(b): $x \cdot 0 = 0$ by duality.

Theorem 3: $(x')' = x$

Theorem 6(a): $x + xy = x.$

Statement	Justification
$x + xy = x \cdot 1 + xy$	postulate 2(b)
$= x(1 + y)$	4(a)
$= x(y + 1)$	3(a)
$= x \cdot 1$	2(a)
$= x$	2(b)

Theorem 6(b): $x(x + y) = x$ by duality

Boolean Functions

- A Boolean function expresses the logical relationship between binary variables.
  - Can be represented in a truth table.
  - The number of rows in the truth table is 2^n, where n is the number of variables in the function.

- The binary combinations for the truth table
  - Obtained from the binary numbers by counting from 0 through 2^n - 1.

- A Boolean function can be transformed from an algebraic expression into a circuit diagram.

There is only one way that a Boolean function can be represented in a truth table.

However, when the function is in algebraic form, it can be expressed in a variety of ways, all of which have equivalent logic.

Example

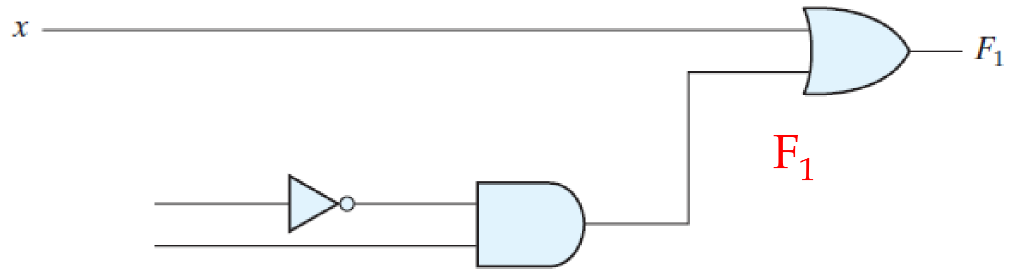
Truth Tables for F_1 and F_2

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

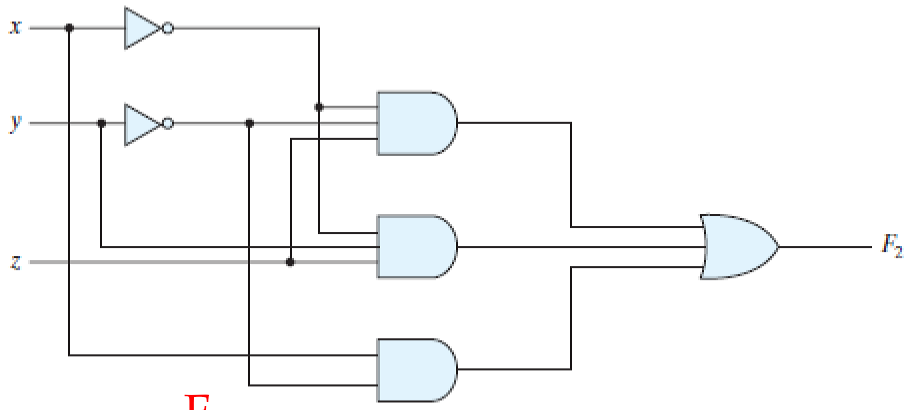
$$F_1 = x + y'z$$

$$F_{2a} = x'y'z + x'yz + xy'$$

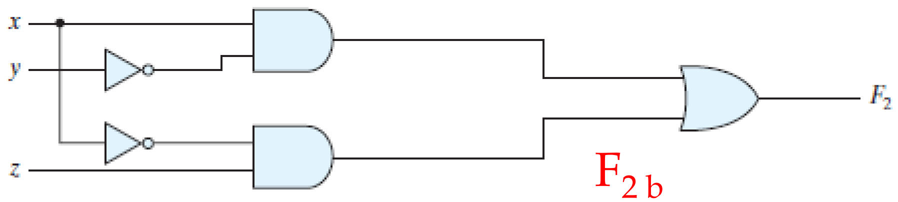
$$\begin{aligned} F_{2b} &= x'y'z + x'yz + xy' \\ &= x'z(y' + y) + xy' \\ &= x'z + xy' \end{aligned}$$



F_1



$F_{2\ a}$



$F_{2\ b}$

- Literal: A single variable within a term in complemented or uncomplemented form.

$F_{2a} = x'y'z + x'yz + xy'$

  - 3 Terms
  - 8 Literals

$F_{2b} = x'z + xy'$

  - 2 Terms
  - 4 Literals

Simplified Examples

1. $x(x' + y) = xx' + xy = 0 + xy = xy.$

2. $x + x'y = (x + x')(x + y) = 1(x + y) = x + y.$

3. $(x + y)(x + y') = x + xy + xy' + yy' = x(1 + y + y') = x.$

4. $$\begin{aligned} xy + x'z + yz &= xy + x'z + yz(x + x') \\ &= xy + x'z + xyz + x'yz \\ &= xy(1 + z) + x'z(1 + y) \\ &= xy + x'z. \end{aligned}$$

5. $(x + y)(x' + z)(y + z) = (x + y)(x' + z)$, by duality from function 4.

Complement of a Function

- DeMorgan’s theorem

De Moran's Theorem

- Use de Moran's Theorem to find complements
  - Sometimes it is more economical to build a circuit using the complement of a function (and complementing its result) than it is to implement the function directly.

- Recall DeMorgan's law states:

$$\overline{(xy)} = \overline{x} + \overline{y} \quad \text{and} \quad \overline{(x+y)} = \overline{x}\,\overline{y}$$

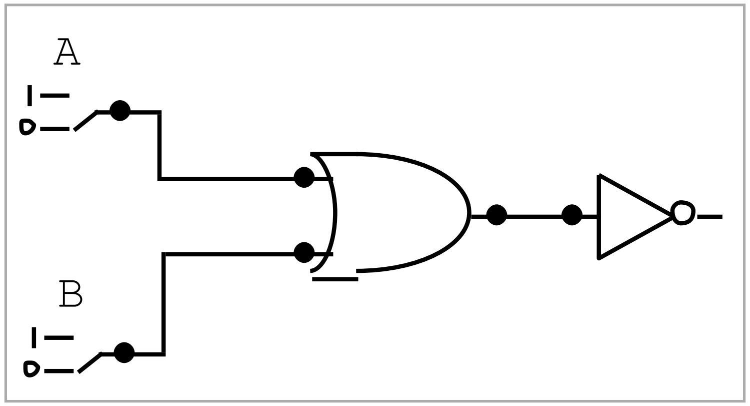
DeMorgan’s Theorem

- DeMorgan’s law provides an easy way of finding the complement of a Boolean function.
- DeMorgan’s theorems states that the complement of a function is obtained by interchanging AND and OR operators and complementing each literal..
- The Demorgan’s theorem mostly used in digital programming and for making digital circuit diagrams.

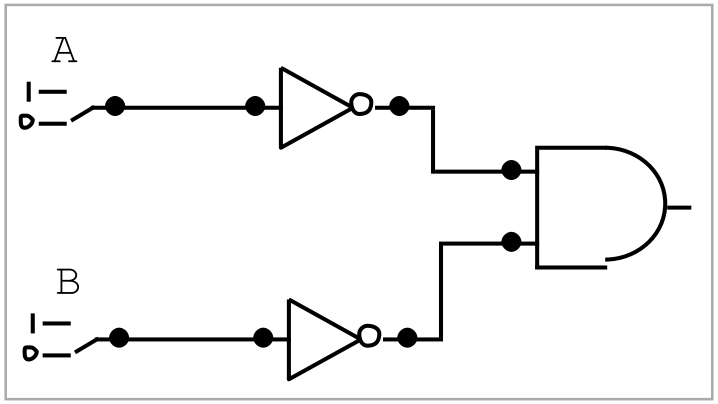
- DeMorgan's 1st Theorem $\overline{A + B} = \overline{A} \cdot \overline{B}$

A	B	A + B	$\overline{A + B}$		$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	0	1		1	1	1
0	1	1	0		1	0	0
1	0	1	0		0	1	0
1	1	1	0		0	0	0

EQUAL



=



Invert output of an OR gate

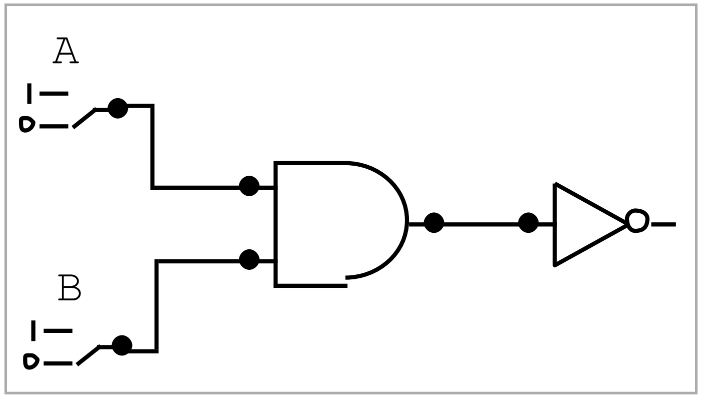
Invert the inputs of an AND gate

- DeMorgan's 2nd Theorem

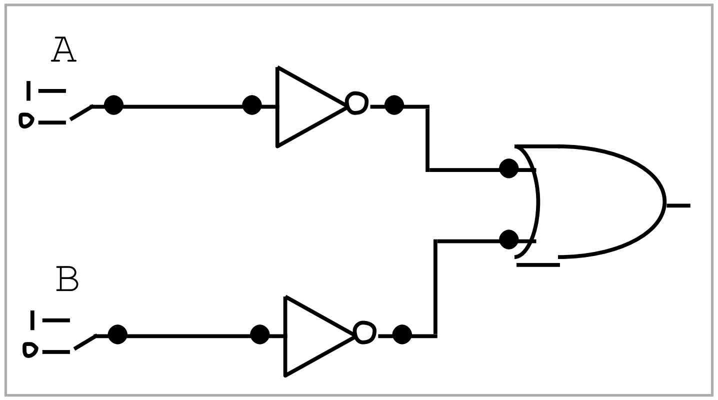
$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

A	B	$A \bullet B$	$\overline{A \bullet B}$		$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	0	1		1	1	1
0	1	0	1		1	0	1
1	0	0	1		0	1	1
1	1	1	0		0	0	0

EQUAL



=



Invert output of an AND gate

Invert the inputs of an OR gate

Example:

$$F = (A + B) \bullet (A' + C), so\ F' = (A' \bullet B') + (A \bullet C')$$

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A	B	C	F′
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Example

- By applying DeMorgan's theorems, find the complement of the functions;

$$F_1 = x'yz' + x'y'z$$

$$F_2 = x(y'z' + yz)$$

- Solution

$$F_1' = (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x + y' + z)(x + y + z')$$

$$\begin{aligned} F_2' = [x(y'z' + yz)]' &= x' + (y'z' + yz)' = x' + (y'z')'(yz)' \\ &= x' + (y + z)(y' + z') \\ &= x' + yz' + y'z \end{aligned}$$

- Find the complement of the functions F_1 and F_2

$F_1 = x'yz' + x'y'z.$
The dual of F_1 is $(x' + y + z')(x' + y' + z).$
Complement each literal: $(x + y' + z)(x + y + z') = F_1'.$

$F_2 = x(y'z' + yz).$
The dual of F_2 is $x + (y' + z')(y + z).$
Complement each literal: $x' + (y + z)(y' + z') = F_2'.$

DeMorgans Theorem in *n* variables

$$\overline{X}\,\overline{Y} = \overline{X+Y}$$

$$\overline{\overline{X}\,\overline{Y}} = X+Y$$

$$\overline{XY} = \overline{X}+\overline{Y}$$

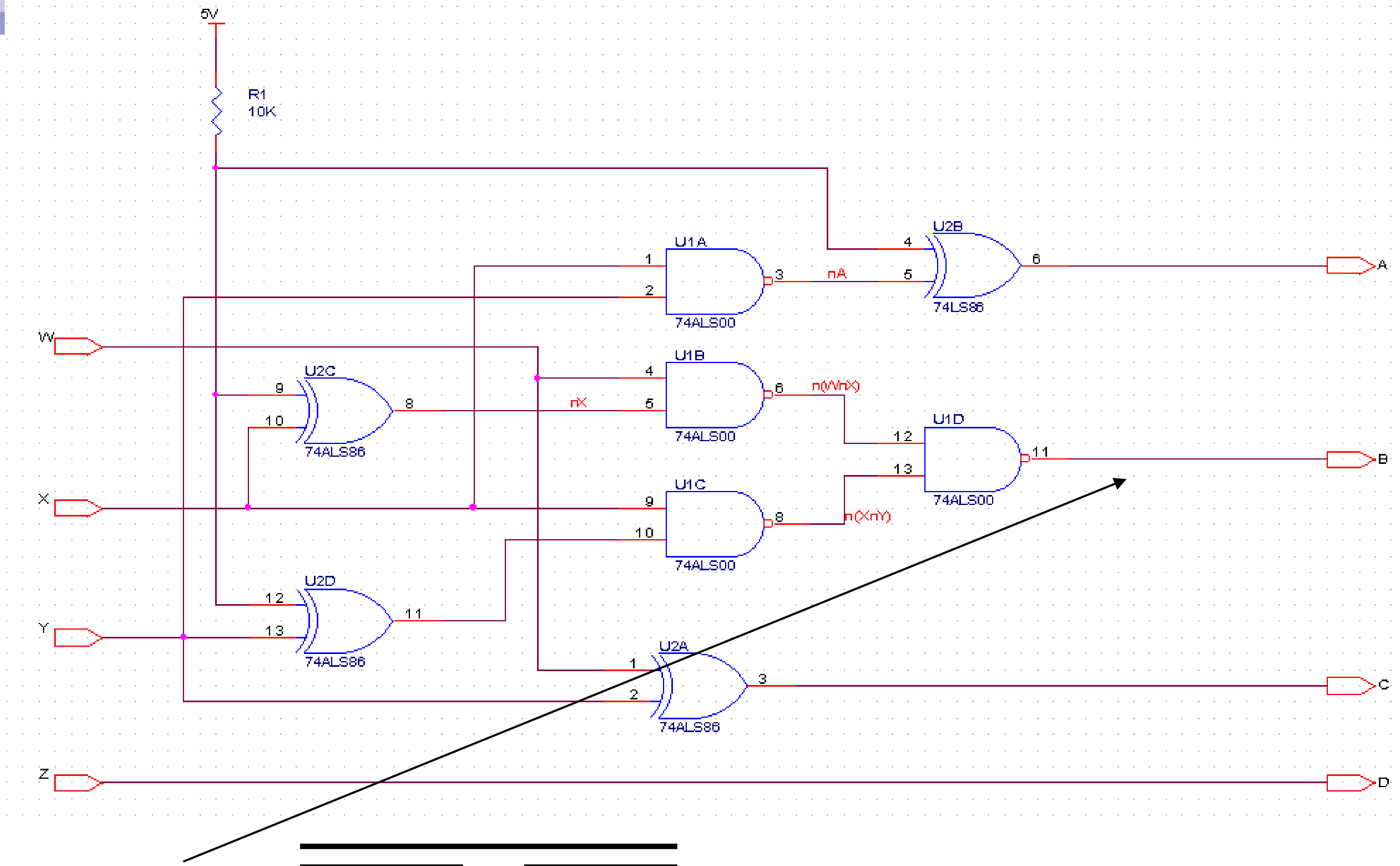
$$XY = \overline{\overline{X}+\overline{Y}}$$

$$\prod_{i=1}^{n} \overline{X_i} = \overline{\sum_{i=1}^{n} X_i}$$

$$\overline{\prod_{i=1}^{n} \overline{X_i}} = \sum_{i=1}^{n} X_i$$

$$\overline{\prod_{i=1}^{n} X_i} = \sum_{i=1}^{n} \overline{X_i}$$

$$\prod_{i=1}^{n} X_i = \overline{\sum_{i=1}^{n} \overline{X_i}}$$



$$B = \overline{\overline{W\bar{X}} \cdot \overline{X\bar{Y}}} = W\bar{X} + X\bar{Y}$$

Boolean Expressions

- In digital design, circuits are expressed as formulae.
- Examples:

3 literals

$$f(x,y,z) = (x' + y)\, z$$

variables

$$f(x,y) = xy' + x'y$$

4 literals

- In general, $f(\mathbf{x})$ is a multi-nomial of degree one in each of the element variables x_k in $\mathbf{x}$.
- The usual goal is to express $f(\mathbf{x})$ in the simplest form possible.
  - Such forms typically involve the minimum number of literals (i.e. variable elements, including complements and repeated elements)
  - They represent hardware designs that are partially or fully optimal.

Canonical and Standard Forms

- Identical functions will have exactly the same canonical form.
  - Minterms and Maxterms
  - Sum-of-Minterms and Product-of- Maxterms
  - Product and Sum terms
  - Sum-of-Products (SOP) and Product-of-Sums (POS)

- When the expression of f is of the form:

  $f = A + BC + DEF$

  SOP

  it is called a Sum of Products, or *Disjunctive Normal Form (DNF).*

- When the expression of f is of the form:

  $f = (A) \,.\, (B + D) \,.\, (D + E + F)$

  POS

  it is called a Product of Sums, or *Conjunctive Normal Form (CNF).*

Shorthand: Σ and Π

In mathematics we encounter SUM and PRODuct of terms and we use the notations:

Greek capital sigma

$$\text{SUM } F_K \Leftrightarrow \sum F_K$$

$$\text{PROD } F_K \Leftrightarrow \prod F_K$$

Greek capital pi

- Product term: literals connected by •
- Sum term: literals connected by +
- Minterm: a product term in which all the variables appear exactly once, either complemented or uncomplemented.
- Maxterm: a sum term in which all the variables appear exactly once, either complemented or uncomplemented.

- Canonical form: Boolean functions expressed as a sum of Minterms or product of Maxterms are said to be in canonical form.

Minterm

- A minterm is a product term in boolean function in which every element is present and is either in normal or in complemented form.
  - Represents exactly one combination in the truth table.
  - Denoted by m_j, where j is the decimal equivalent of the minterm's corresponding binary combination (b_j).
  - A variable in m_j is complemented if its value in b_j is 0, otherwise is uncomplemented.

  - If value is 0 then we take the complement of the variable.
    If value is 1 then we take the variable as is.

Minterm Steps

i. Write the term consisting of all the variables
ii. Replace all complement variables like x or x' with 0
iii. Replace all non-complement variables like x or y with 1
iv. Express the decimal equivalent of the binary formed in the above steps

- The decimal number is then written as a subscript of letter **m** where, small m denote minterm.

Example:

- Assume 3 variables (A, B, C), and j=3. Then, b_j = 011 and its corresponding minterm is denoted by m_j = A'BC

Maxterm

- **A maxterm** is a **sum** term in boolean function in which every element is present and is either in normal or in complemented form.
  - Represents exactly one combination in the truth table.
  - Denoted by M_j, where j is the decimal equivalent of the maxterm's corresponding binary combination (b_j).
  - A variable in M_j is complemented if its value in b_j is 1, otherwise is uncomplemented.
  - **If value is 1 then we take the complement of the variable.**
    If value is 0 then we take the variable as is.

Maxterm Steps

i. Write the term consisting of all the variables
ii. Replace all complement variables like x or x' with 1
iii. Replace all non-complement variables like x or y with 0
iv. Express the decimal equivalent of the binary formed in the above steps

- The decimal number is then written as a subscript of letter **M** where, capital M denote maxterm.

Example:

- Assume 3 variables (A, B, C), and j=3. Then, $b_j = 011$ and its corresponding maxterm is denoted by $M_j = A+B'+C'$

Truth Table notation for Minterms and Maxterms

			Minterms		**Maxterms**	
x	***y***	***z***	**Term**	**Designation**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

- Examples

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

- Consider a Truth table for $f_1(a,b,c)$ at right
- The canonical sum-of-products form for f_1 is

$$f_1(a,b,c) = m_1 + m_2 + m_4 + m_6$$

$$= a'b'c + a'bc' + ab'c' + abc'$$

- The canonical product-of-sums form for f_1 is

$$f_1(a,b,c) = M_0 \bullet M_3 \bullet M_5 \bullet M_7$$

$$= (a+b+c)\bullet(a+b'+c')\bullet (a'+b+c')\bullet(a'+b'+c').$$

a	b	c	f_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

- maxterm with subscript *j* is a complement of the minterm with the same subscript *j* and vice versa.

- Homework
  - Sum of Minterms
  - Product of Maxterms
  - Conversion between Canonical Forms

Conversion of SOP from standard to canonical form

Example

- Express the Boolean Function F = A+B’C as a sum of minterms

Conversion of SOP from standard to canonical form

Example

Express the Boolean function F = A + B′C as a sum of minterms.

Solution: The function has three variables: A, B, and C. The first term A is missing two variables; therefore,

$$A = A(B + B') = AB + AB'$$

This function is still missing one variable, so

$$A = AB(C + C') + AB'(C + C')$$
$$= ABC + ABC' + AB'C + AB'C'$$

The second term B′C is missing one variable; hence,

$$B'C = B'C(A + A') = AB'C + A'B'C$$

Combining all terms, we have

$$F = A + B'C$$
$$= ABC + ABC' + AB'C + AB'C' + A'B'C$$

But AB′C appears twice, and according to theorem (x + x = x), it is possible to remove one of those occurrences. Rearranging the minterms in ascending order, we finally obtain

$$F = A'B'C + AB'C + AB'C + ABC' + ABC$$
$$= m1 + m4 + m5 + m6 + m7$$

When a Boolean function is in its sum-of-minterms form, it is sometimes convenient to express the function in the following brief notation:

$$F(A, B, C) = \sum m\,(1, 4, 5, 6, 7)$$

Conversion of SOP from standard to canonical form

Example

- Express the Boolean Function $f = xy + x'z$ as a sum of maxterms

Example

Express the Boolean function $F = xy + x'z$ as a product of maxterms.

Solution: First, convert the function into OR terms by using the distributive law:

$F = xy + x'z = (xy + x')(xy + z)$

$= (x + x')(y + x')(x + z)(y + z)$

$= (x' + y)(x + z)(y + z)$

The function has three variables: x, y, and z. Each OR term is missing one variable; therefore,

$x' + y = x' + y + zz' = (x' + y + z)(x' + y + z')$

$x + z = x + z + yy' = (x + y + z)(x + y' + z)$

$y + z = y + z + xx' = (x + y + z)(x' + y + z)$

Combining all the terms and removing those which appear more than once, we finally obtain

$F = (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z)$

F= M0M2M4M5

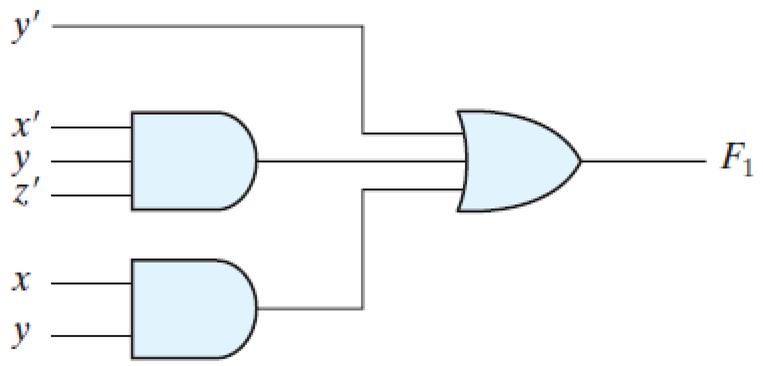
A convenient way to express this function is as

follows: $F(x, y, z) = \pi M(0, 2, 4, 5)$

The product symbol, π, denotes the ANDing of maxterms; the numbers are the indices of the maxterms of the function.

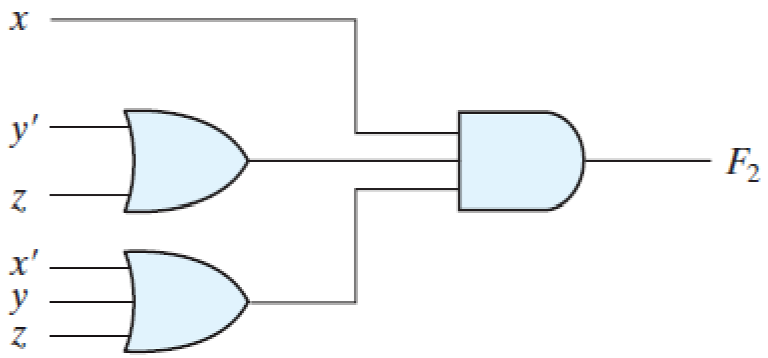
Standard Forms

$$F_1 = y' + xy + x'yz'$$



(a) Sum of Products

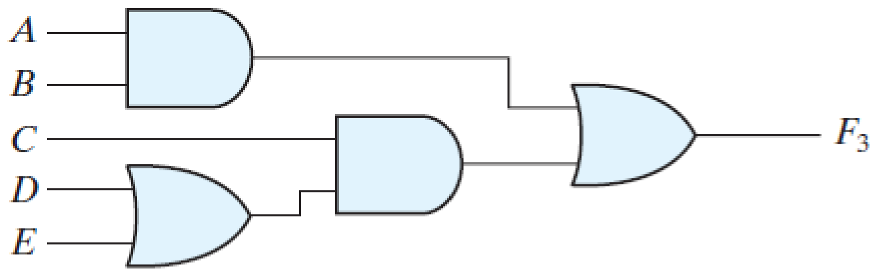
$$F_2 = x(y' + z)(x' + y + z')$$



(b) Product of Sums

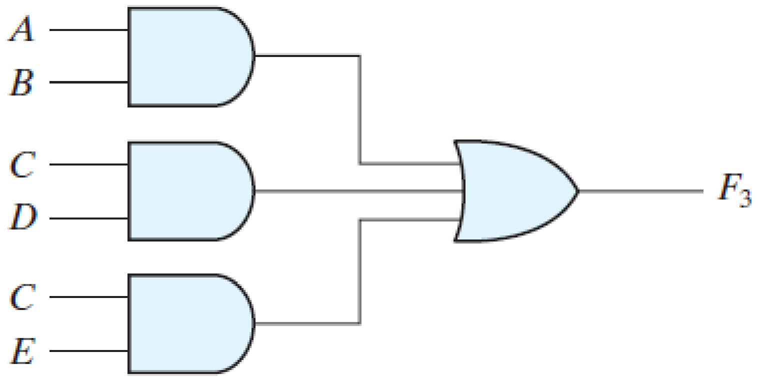
$F_3 = AB + C(D + E)$

(nonstandard form)



$$F_3 = AB + C(D + E)$$
$$= AB + CD + CE$$

(standard form)



Integrated Circuit

- Fabricated on a die of a silicon semiconductor crystal - a *chip*
- *Chip* - contains electronic components for constructing digital gates.
  - The various gates are interconnected inside the chip to form the required circuit.
- Each IC has a numeric designation printed on the surface of the package for identification.

Levels of Integration

- *Small-scale integration* (SSI)
  - Several independent gates in a single package.
  - Inputs and outputs of the gates are connected directly to the pins in the package. The number of gates = fewer than 10
- *Medium-scale integration* (MSI)
  - Approximately 10 to 1,000 gates in a single package.
  - Perform specific elementary digital operations.
  - E.g. decoders, adders, and multiplexers, registers and counters.
- *Large-scale integration* (LSI)
  - Contain thousands of gates in a single package.
  - E.g. processors, memory chips, and programmable logic devices.
- *Very large-scale integration* (VLSI)
  - Contain millions of gates within a single package.
  - E.g. large memory arrays and complex microcomputer chips.

Summary of Boolean Expressions

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

Design Example

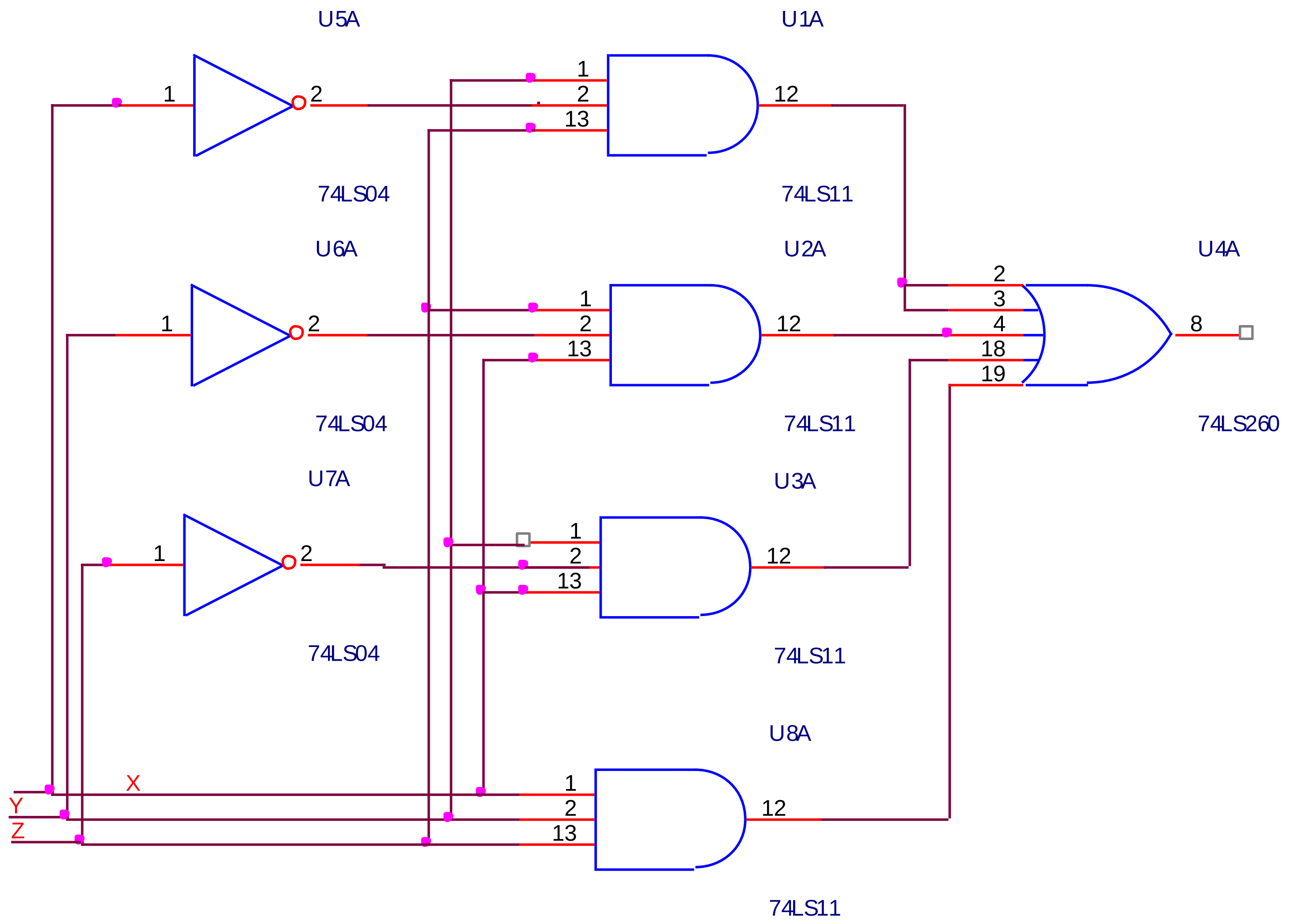
- Design a 3-input majority circuit given by the function

$$F = \overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + XYZ$$

X	Y	Z	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

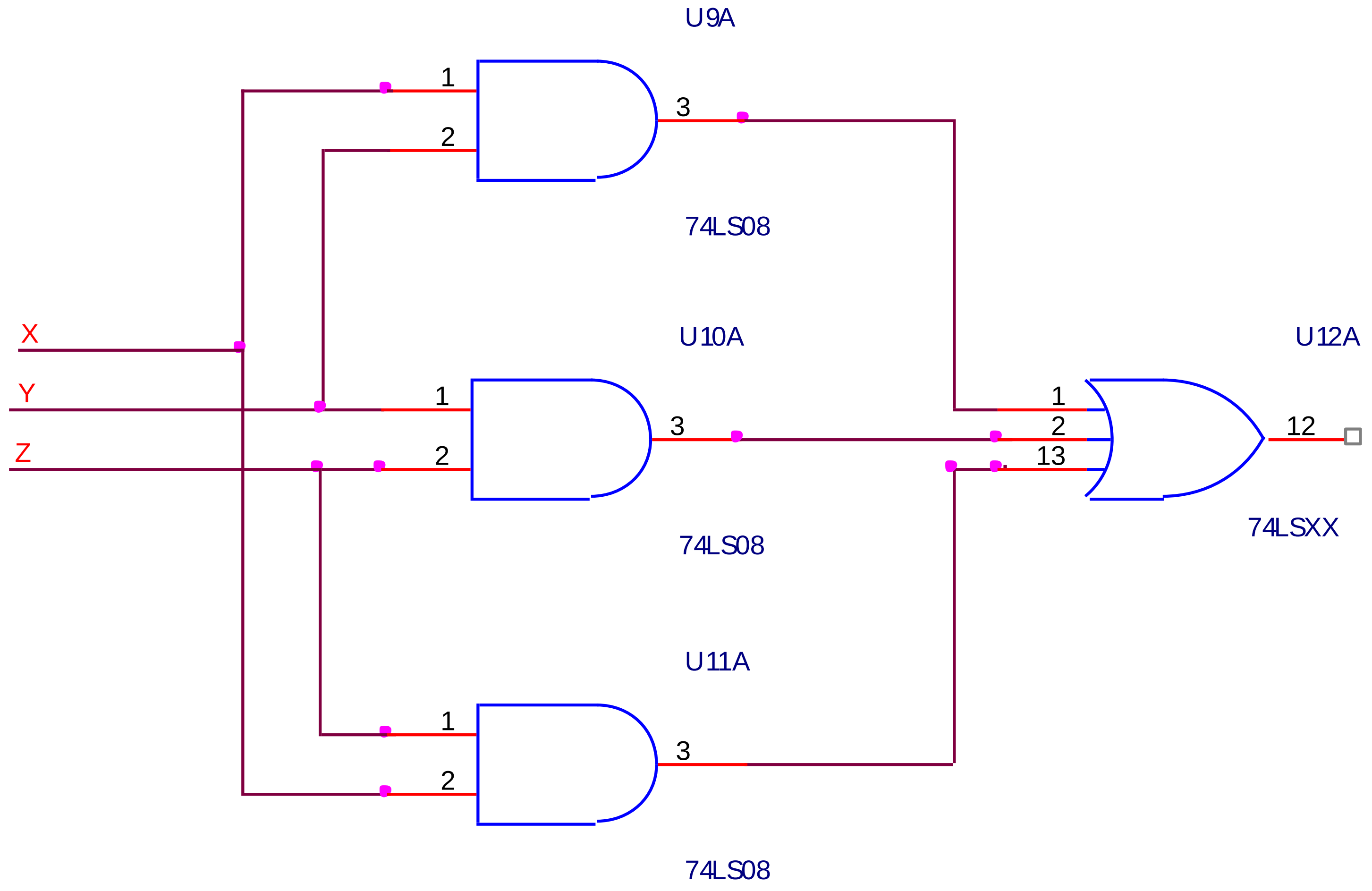
$$F = \overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + XYZ$$



Simplified Version

$$F = \overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + XYZ = YZ + XZ + XY$$

$$\begin{aligned}
F &= \overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + XYZ \\
&= \overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + XYZ + XYZ + XYZ \\
&= YZ(\overline{X} + X) + XZ(\overline{Y} + Y) + XY(\overline{Z} + Z) \\
&= YZ \cdot 1 + XZ \cdot 1 + XY \cdot 1 \\
&= YZ + XZ + XY
\end{aligned}$$


U9A
1
2
3
74LS08
X
Y
Z
U10A
1
2
3
74LS08
U11A
1
2
3
74LS08
U12A
1
2
13
12
74LSXX

Example

- Show how to build an 8 input and gate using several two input and gates.
- Which is better? Why?